smart_data_preparation

October 16, 2020

1 S.Ma.R.T. DATA PREPARATION

[1]:
```
# Se define el estilo de visualización del notebook:
from IPython.core.display import display, HTML
display(HTML("<style>.container { width:90% !important; }</style>"))
```

```
<IPython.core.display.HTML object>
```

[2]:
```
import time
from datetime import timedelta
start_time = time.time()
```

[3]:
```
import datetime as dt
import numpy    as np
import pandas   as pd
import scipy

extrae_id = lambda x : (x.strip('ObjectId').strip('("')'))

def remove_outliers_6sigma(X):
    return X[abs(X - np.mean(X)) < 3 * np.std(X)]
```

[4]:
```
daylist         = ["all_data.csv", "20200124.csv", "20200201.csv", "20200202.
 ↪csv", "20200203.csv", "20200204.csv"]
filename_input  = daylist[0]
ruta            = "./dataset/" + filename_input
filename_output = "./dataset/log_{}".format(filename_input)
df              = pd.read_csv(ruta, converters = {'_id':extrae_id , 'DrillID':
 ↪extrae_id})
```

[5]:
```
df.sort_values(['TagID', 'DrillID', 'TimeStamp'], ascending=[True, True, True],␣
 ↪inplace=True)

if df.duplicated().sum() > 0:
    df = df.drop_duplicates()
```

[6]:
```
df.drop(list(df.columns[df.apply(pd.Series.nunique)==1]), axis=1, inplace=True)

if df.isnull().sum().sum() > 0:
    df = df.fillna(0)
```

[7]:
```
df['PositionX']   = pd.to_numeric(df['Position'].str.strip('[]').str.
 ↪split(pat=",", expand = True)[0], downcast='float')
df['TimeStampDT'] = pd.to_datetime(df['TimeStamp'], unit='ms')
df                = df.drop(['Position'], axis=1)
```

[ ]:
```
df["diff_posx_m"]   = df["PositionX"].diff().abs()
df["diff_time_sec"] = df["TimeStamp"].diff().abs() / 1000
df["LinearSpeed"]   = df["diff_posx_m"].div(df["diff_time_sec"])
df["Acceleration"]  = df["diff_posx_m"].div(df["diff_time_sec"]**2)

if df.isnull().sum().sum() > 0:
    df = df.fillna(0)
```

[ ]:
```
df          = df.drop( (df[df['LinearSpeed']  > 1.56].index ) )
df          = df.drop( (df[df['Acceleration'] > 0.03].index ) )

variables   = ["diff_posx_m", "diff_time_sec", "LinearSpeed", "Acceleration"]

df_aux1     = pd.merge(df     , pd.
 ↪DataFrame(remove_outliers_6sigma(df[variables[0]].values)), how = 'left',
                        left_index = True, right_index = True).rename(columns={0:
 ↪ variables[0] + '_no_outlier'})
df_aux2     = pd.merge(df_aux1, pd.
 ↪DataFrame(remove_outliers_6sigma(df[variables[1]].values)), how = 'left',
                        left_index = True, right_index = True).rename(columns={0:
 ↪ variables[1] + '_no_outlier'})
df_aux3     = pd.merge(df_aux2, pd.
 ↪DataFrame(remove_outliers_6sigma(df[variables[2]].values)), how = 'left',
                        left_index = True, right_index = True).rename(columns={0:
 ↪ variables[2] + '_no_outlier'})
df_filtered = pd.merge(df_aux3, pd.
 ↪DataFrame(remove_outliers_6sigma(df[variables[3]].values)), how = 'left',
                        left_index = True, right_index = True).rename(columns={0:
 ↪ variables[3] + '_no_outlier'})

df_filtered.dropna(inplace=True)
df_filtered.drop(variables, axis=1, inplace=True)
df_filtered.rename(columns={"diff_posx_m_no_outlier"  : "diff_pos",
                            "diff_time_sec_no_outlier": "diff_time",
                            "LinearSpeed_no_outlier"  : "Speed",
                            "Acceleration_no_outlier" : "Accel"}, inplace=True)
```

```
[ ]: df_agg = df_filtered.groupby(['TagID', 'DrillID']).agg({'TimeStamp'  :␣
     ↪['min','max'],
                                                             'TimeStampDT':␣
     ↪['min','max'],
                                                             'PositionX'  :␣
     ↪['min','max', 'mean'],
                                                             'diff_pos'   :␣
     ↪['min','max', 'mean'],
                                                             'diff_time'  :␣
     ↪['min','max', 'mean'],
                                                             'Speed'      :␣
     ↪['min','max', 'mean'],
                                                             'Accel'      :␣
     ↪['min','max', 'mean']}).reset_index()
```

```
[ ]: df_agg['Outlier_PositionX'] = ( df_agg['PositionX']['max'] *␣
     ↪df_agg['PositionX']['min'] ) >= 0
     df_mask_X                   = df_agg['Outlier_PositionX']==False
     filtered_X                  = df_agg[df_mask_X]
     df_agg                      = filtered_X.drop('Outlier_PositionX', axis=1)
     df_agg.columns              = ['_'.join(col_).strip() for col_ in df_agg.
     ↪columns.values]
     df_agg['classifier_']       = np.log( (1 + df_agg['Speed_mean']) / ( 1 +␣
     ↪(df_agg['Accel_mean'] * df_agg['diff_time_mean']) ) )
```

```
[ ]: df_agg.drop(['diff_time_max', 'Speed_mean'], axis=1, inplace=True)
     df_agg.to_csv(filename_output, index=False)
```

```
[ ]: elapsed_time_secs = time.time() - start_time
     msg = "Execution took: %s secs (Wall clock time)" %␣
     ↪timedelta(seconds=round(elapsed_time_secs))
     print(msg)
```

```
[ ]: !jupyter nbconvert --to pdf smart_data_preparation.ipynb
```

```
[ ]:
```